(12)

Semiannual Technical Summary

Research in the VLSI Program

30 September 1981

Lincoln Laboratory

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3797).

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

RESTRUCTURABLE VLSI PROJECT

SEMIANNUAL TECHNICAL SUMMARY REPORT TO THE DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

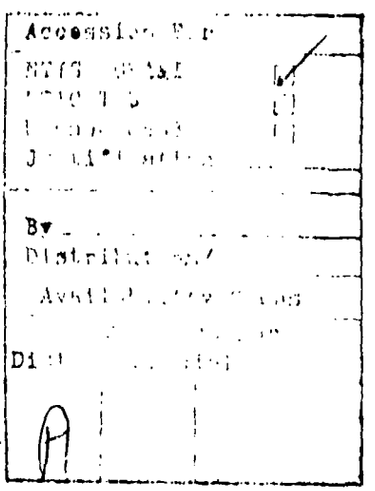
1 APRIL — 30 SEPTEMBER 1981

ISSUED 14 JANUARY 1982

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes work on the Restructurable VLSI Research Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the semiannual period 1 April through 30 September 1981.



CONTENTS

RESTRUCTURABLE VLSI PROGRAM

I. PROGRAM OVERVIEW AND SUMMARY

The main objective of the Lincoln Restructurable VLSI Program (RVLSI) is to develop methodologies and architectures for implementing whole-wafer-scale systems with complexities approaching a million gates. In our approach, we envisage a modular style of architecture comprising an array of cells embedded in a regular interconnection matrix. Ideally, the cells should consist of only a few basic types. The interconnection matrix is a fixed pattern of metal lines augmented by a complement of programmable switches or links. Conceptually, the links could be either volatile or nonvolatile. They could be of an electronic nature such as a transistor switch, or could be permanently programmed through some mechanism such as a laser. A present major thrust of the current RVLSI Program is toward laser-formed interconnect.

The link concept offers the potential for a highly flexible, restructurable type of interconnect technology which could be exploited in a variety of ways. For example, logical cells or subsystems found to be faulty at wafer probe time could be permanently excised from the rest of the wafer. The flexible interconnect could also be used to "jump around" faulty logic and tie in redundant cells judiciously scattered around the wafer for this purpose. Also, the interconnect could be tailored to a specific application in order to minimize electrical degradations and performance penalties caused by unused wiring.

Further, the testing of a particular logical subsystem buried deep within a complex wafer-scale system is a very difficult problem. A properly designed restructurable interconnect matrix could be temporarily configured to render internal cells both controllable and observable at the wafer periphery. In this way, each component cell or a tractable cluster of cells could be tested in a relatively straightforward way using standard techniques.

With an electronic type of linking mechanism it is possible to think in terms of a dynamically reconfigurable system. Such a feature could be used to alter the functional mode of a system subject to changes in the operating

scenario, or it could be used to support some degree of fault tolerance if the system architecture were suitably designed.

Several major areas of research have been identified in the context of the RVLSI concept:

(a) System architectures and partitionings for whole-wafer implementations.

(b) Placement and routing strategies for optimal utilization of redundant resources and efficient interconnect.

(c) Assignment and linking algorithms to exploit redundancy and flexible interconnect.

(d) Methods for expediting cell design with emphasis on functional level descriptions, enhanced testability, and fault tolerance.

(e) Methods for testing complex, multiple-cell, whole-wafer systems.

Complementary work on the development of various link and interconnect technologies as well as fabrication/processing technology is being supported by the Lincoln Air Force Line Program, and results are reported under the Lincoln Laboratory Advanced Electronic Technology Quarterly Technical Summary.

Work for this period is reported under the general headings of RVLSI Technology (Sec. II), Design Aids for RVLSI Systems (Sec. III), and RVLSI Testing and Applications (Sec. IV). In the area of RVLSI technology, test structures designed to investigate the possibility of laser programmable links for MOSIS have been evaluated. It is concluded that acceptable link quality is not practical using the present MOSIS process and steps are being taken to provide the MOSIS community with two-layer metal and quality laser programmable links through the Lincoln polyimide/amorphous silicon process which has been successfully demonstrated in the Lincoln semiconductor processing facility.

Under the heading of Design Aids for RVLSI Systems, the MACPITTS silicon compiler implementation is progressing on schedule with only three components remaining to be designed and coded. A production-oriented

assignment/linking/laser control system is in place, optimized for the packet radio integrator application (PAL 81). A wafer description language was developed to describe interconnect, a an interface to the laser control microprocessor was implemented. Also, a new assigner suitable for arbitrary bus segmentation was developed. Recoding of the mask design rule checker (MDRC) on the VAX is nearly complete with 23 mask data-processing primitives in place. The system has been successfully exercised with the 3-4-5 speech filter CIF. A preliminary theory of device cost based on active area, redundancy, yield, interconnect area, and algorithm execution time has been developed. Analyses of some popular interconnect strategies have shown them to be impractical for highly dense, large-area systems. A yield model which makes some accounting of the area occupied by interconnect circuitry has been developed, and a variety of simulations have been conducted.

With regards to Applications and Testing, Phase 0 packet radio integrator wafers have been fabricated. Cell yields are sufficient to proceed with checkout of the PAL 81 system and the laser assembly. A packaging technique adequate for the current fabrication and testing sequences has been devised, and prototype packages fabricated. Connections to testing equipment have been provided. The Phase I Integrator 4- × 10-bit counter cell, complete with multiplexers and bus drivers, has been completely specified, designed, and submitted to the Lincoln bulk CMOS process. Present plans call for first silicon by mid-December 1981. Some effort has been directed toward the physical layout of the total wafer and interconnect grid, but is as yet incomplete. A modification to the Mann 4800 Direct-Step-On-Wafer (DSW) machine has been devised and is nearing completion, which will greatly reduce the number of reticles required.

The VAX facility has been augmented with a pair of SI disks, and UNIX (4.1 BSD) has been installed. The Berkeley CAESAR system has been successfully integrated using a borrowed AED 512 terminal. All hardware for a VAX/ARPANET connection is in place, and the best method for obtaining needed software is under investigation.

II. RVLSI TECHNOLOGY

MOSIS LASER-FORMED CONNECTION EXPERIMENT

Redundancy is necessary in very large (wafer-scale) integrated circuits, since processing is imperfect and a certain percentage of the circuitry will not be functional. One approach to the defect avoidance problem is to partition the total circuit into pieces which can be individually tested after fabrication. These pieces are then interconnected using an X-Y grid of conductors. Primary interest centers on the device placed at the vertical-horizontal crossings in the grid. Although a wide variety of possibilities are available, laser zappable links look very promising and have many desirable characteristics such as low "on" resistance, high "off" resistance, visually checkable connections, high reliability, and inexpensive programming equipment.

The DARPA MOSIS process provides only single-level metal, whereas the zappable links built at Lincoln Laboratory use metal-metal structures.[1] Data gathered at Lincoln indicated that there were failure modes involving metal-poly and metal-substrate shorts in various test structures. Thus, the possibility was present that these mechanisms could be exploited to form useful links in the MOSIS context.

A test chip was designed for MOSIS fabrication comprising nine 3-by-3 test arrays of metal-poly links and a similar number of metal-diffusion links. The dimensions of the test links were very nearly the same as those used in conjunction with the Lincoln bulk CMOS facility, making the same probe card and test equipment usable. Many copies of the chip have been fabricated and received at Lincoln Laboratory.

To date, no high-quality MOSIS contacts have been established. Electrical characterizations of the links have revealed I-V characteristics with zener diode-like behavior. Typical curves for metal-polysilicon and metal-diffusion structures are shown in Figs. 1(a) and (b), respectively. Measured resistances seem to be on the order of a few hundred ohms. Yield behavior is also relatively poor for any laser power setting, as shown in Tables I and II. SEM studies of programmed links indicate very erratic structural behavior and production of much undesirable debris.

It is tentatively concluded that the simplistic approach thus far taken will not prove viable. Steps are currently being taken to provide a second-level metal and laser formable link capability for the MOSIS community through the Lincoln polyimide insulator process. Initial experiments with partially processed MOSIS test chips are planned for the first and second quarters of FY 82.

TABLE I

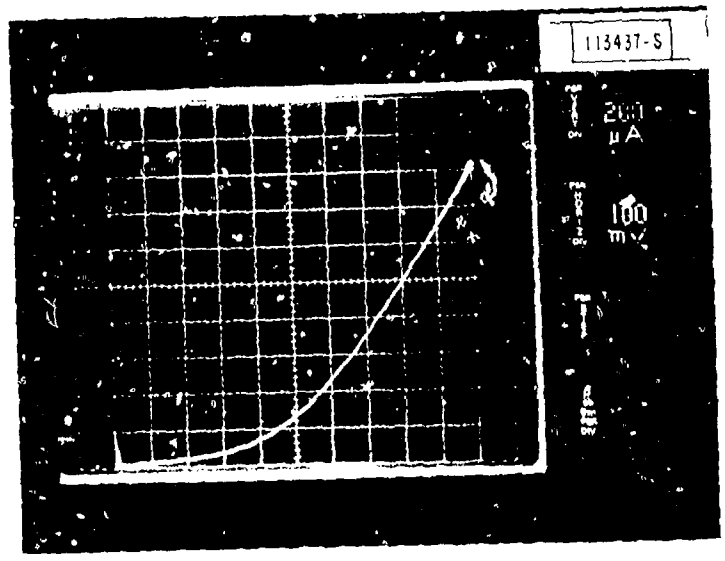
YIELD VS LASER POWER FOR METAL-POLYSILICON LINKS

Laser Power (W)	Yield (percent)
2.0	12
2.5	30
3.0	20
3.5	0
4.0	0
4.5	0
5.0	40

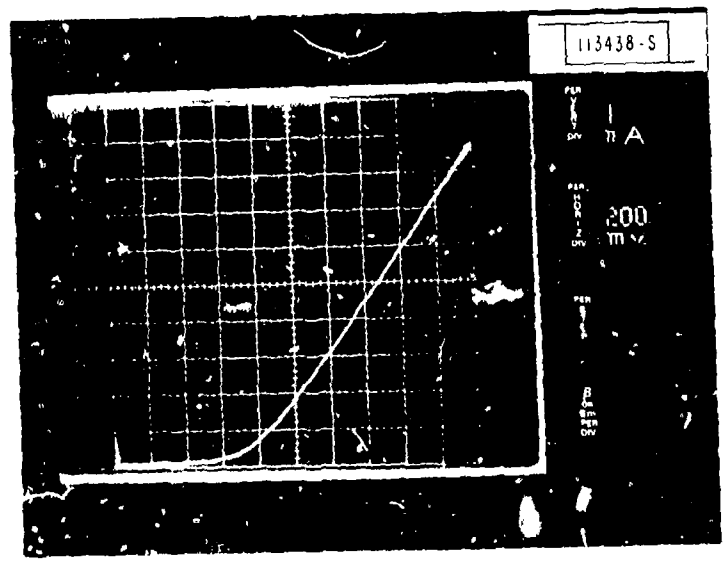
TABLE II

YIELD VS LASER POWER FOR METAL-DIFFUSION LINKS

Laser Power (W)	Yield (percent)
3.75	16
4.0	0
4.25	20
4.5	10
4.75	20
5.0	10



(a) Metal-polysilicon (R ≈ 330 Ω).



( ) Metal-diffusion (R ≈ 130 Ω).

Fig. 1. Typical I-V characteristics for MOSIS links.

III. DESIGN AIDS FOR RVLSI SYSTEMS

A. CELL DESIGN

An integrated circuit design may be specified at various levels of detail. Describing a design at the lower levels of detail – namely the geometric, topological, or connective levels – has many drawbacks. Design at these levels is tedious, time consuming, inflexible, and very error prone. In addition, circuit specification at lower levels removes much semantic design information which could aid automatic modification of the design. Despite these shortcomings, many designers prefer to pay the price and handcraft a design for maximum speed and area efficiency due to the lack of available alternatives.

Many designs fit into the framework of microprogram sequenced data-path operations. For this large class of systems, the MACPITTS silicon compiler represents an alternative to handcrafted design. A flexible register transfer-type language was developed particularly suited to the capabilities available through custom VLSI design. Features of this language include multiple way branching, nested conditions of arbitrary Boolean expressions, subroutine capability, and parallel processes. A compiler was developed which converts programs in this language directly to mask-level specifications. MACPITTS is being used to implement a minimal dynamic circuit tester as an initial vehicle for proving its capability.

The language and compiler have several mechanisms for producing efficient, high-performance designs. First, a custom data path is constructed to exactly meet the requirements of the source functional description. This data path has operators intermixed with registers, and allows simultaneous operations and transfers on an arbitrary number of parallel buses. Second, the compiler uses a unique heuristic to normalize the finite state control into a multilevel sum-of-products notation for efficient PLA factorization. Third, the language allows specifying that several parallel tasks occur simultaneously on a chip which can communicate with each other either through shared registers or by signals. Conventional hardware designs typically exploit large amounts of parallelism which is necessary for achieving high throughput in

many integrated circuit application areas such as digital-signal processing. Despite this, most silicon compiler designs proposed so far impose a sequential flavor, borrowed from the software domain, on the hardware designs handled by the compiler. The above combination of mechanisms seems necessary for any efficient silicon compiler, but is apparently unique among all the known compilers available or proposed. The Baekus-Naur Form (BNF) description for MACPITTS is given in Appendix A.

The functional level of specification offers several advantages over lower levels. First, resulting designs need not be checked for design rule violations, as the compiler generates designs which are correct by synthesis. Second, the time-consuming steps of node extraction and switch-level simulation can be replaced with a higher-level, faster functional simulation. To date, such a simulator has been constructed which accepts the intermediate output of the compiler, aiding the debugging of the compiler as well as the debugging of designs. Finally, the compiler can be extended to generate a variety of alternative designs in the designated space equivalent to the same functional description. This will allow future insertion of aids for automatic test vector generation, circuit testability, self-test capability, defect avoidance, and fault tolerance. Also, the modular nature of the synthesis technique provides a framework for the development of automatic system partitioning aids which makes MACPITTS particularly attractive for the cell-oriented RVLSI design environment.

The MACPITTS compiler transforms a functional circuit specification into a geometric layout implementation. The source-code input is a register transfer-level description of the desired system. The target architecture for implementing the system is a combination of finite-state machines, one for each of the parallel processes in the source code, and a data-path unit. The finite-state machines are implemented as generalized multilevel PLAs with a sequential next-state unit attached. The compiler consists of two levels of routines. The higher-level routines examine the source code and extract a technology-independent, intermediate-level description of the system in terms of data-path specifications, control equations, and state assignments. The lower-level routines bind this intermediate-level description to an actual

mask layout. The output of the lower-level routines is actual CIF code specific to the MOSIS/NMOS process. Following is a concise description of the major components of the MACPITTS system.

1. Prepass

All the syntax checks of the compiler are combined into one set of routines. This eases the task of writing and verifying the rest of the compiler by allowing it to assume that it has received a syntactically correct program. This pass also implements a macro expansion feature which makes the MACPITTS language somewhat extensible.

2. Data-Path Extraction

This pass scans the source code to determine the set of registers and operators needed to implement the required data path. It understands the parallel semantics of certain MACPITTS constructs which allow it to optimize the data path. When segments of code have sequential or mutually exclusive execution, the operators required by those code segments can be shared. Code segments which are to be executed in parallel will cause the necessary duplication of operators in the data path. The output of this pass is a technology-independent data-path specification in a well-defined format.

3. State Extraction

The target architecture for MACPITTS specifies finite-state machines which have extended counters as state registers to allow a simple program-like sequential control flow. State assignment is thus constrained to be linear and is therefore straightforward. This pass counts the number of states in each process and builds a symbol table of "go" labels. It outputs a specification of the width and stack depth required of the next state units for each process in a technology-independent manner.

4. Control Extraction

This pass converts the source code into Boolean equations which describe the interaction between the data path, the next-state units, and the outside

world. It accepts a rather general input format which allows nested conditionals, arbitrary Boolean conditions, and complex formulas. The output is, again, technology-independent.

5. Normalizer

Conventional PLAs used to implement Boolean equations require those equations to be in strict sum-of-products notation. Many designs would create excessively large PLAs when constrained to this format. Instead, MACPITTS normalizes the equations into a multiple-level AND/OR network whose depth is heuristically controlled to limit combinatorial explosion. It is intended that such multiple-level AND/OR planes be implemented either as extended PLAs or as Weinberger-type gate matrices.[2]

6. Data-Path Layout

This pass commits the data-path specification previously extracted from the source program to a geometric layout. These custom-generated data paths have registers and operators intermixed along several shared internal buses. This unique architecture allows several operations and data transfers to occur in the data path on different buses during the same clock cycle. The components of the data path have custom multiplexers on their inputs which implement exactly those data-transfer capabilities required by the source program.

7. Next-State-Unit Layout

This pass builds geometric layouts for the next-state units needed for each process in the source program. These next-state units provide a default next state to the finite-state machine in the absence of a "go" command, and thus support conventional sequential program flow. An optional state stack is used for subroutine calls and returns.

8. PLA Layout

As mentioned before, efficient implementation of the control equations requires multiple-level AND/OR planes. This set of routines generates the layout of such PLAs.

9. Channel Router

This pass connects together the various parts of the complete system. This task is simplified since all components are constrained to have their pins on a single side and connect to a single linear channel.

10. Organelle Library

The Data-Path Layout program references this organelle library to determine the layout and characteristics of each primitive operator. The term "organelle" refers to a single bit slice of a register or operator. The Data-Path Layout program can be augmented with new operators simply by adding new definitions to the library which has a well-defined format. The basic initial library at present includes the organelles for: and, or, not, nand, nor, xor, equ, 1+, 1–, +, –, =, <, <<, >>.

11. L5

This is a Lisp-based geometric IC layout language used by all the layout routines to construct CIF files. It bears a strong resemblance to LICL, a similar language implemented in C (see Ref. 3). A prime attribute of L5 is its total applicative nature.

12. Simulator

This pass takes the intermediate-level technology-independent code from the compiler and performs a functional simulation. It simulates the parallel execution of processes as well as the data-path operations. The simulator can be extended to handle new data-path operators by including a simulation form with each new organelle added to the library. Semantic error checking is provided in the simulator to aid the verification of designs before fabrication.

At present, all sections of MACPITTS are operational except the Data-Path Layout Module, Next-State Unit Layout Module, PLA Layout Module, Channel Router, and Simulator. Of these sections, the Data-Path Layout, Next-State Unit Layout, and Simulator have been written and are in the final debugging stages. The PLA Layout and Channel Router have not yet been

coded. However, since other versions of similar functionality have been written in the past, implementation of these modules is expected to be straightforward.

B. PRODUCTION-ORIENTED ASSIGNMENT/LINKING SYSTEM

In the previous semiannual,[4] the research-oriented Placement, Assignment, and Linking System (PAL) was described in great detail. The description concluded with a brief overview of the proposed production-oriented PAL system. Due to early anticipated use, a simplified production-oriented PAL (PAL 81) was conceived. PAL 81 is to be specifically oriented to laser-link technology and the integrator problem. The assign and link software will take this restriction into account, and therefore be somewhat more efficient than general approaches. However, the interface formats for the various parts of the PAL system will be capable of handling more general cases. In addition the PAL 81 software will capitalize on the advantages of arbitrary segmentation offered by the laser link technology.

Specifically, PAL 81 is intended to automate the assignment and linking process working from a "bad cell" map to produce a fully connected integrator system. This implies that the PAL system must also control the laser table assembly. Appendix B contains a description of the VAX/Laser Controller interface. The elements of the system are shown in Fig. 2. All the modules on the VAX can be easily exchanged for different or more powerful versions of the same function. The modularity of the system will make it adaptable to any other application that uses laser-based restructurability.

To perform assignment and linking on a real wafer, three types of information must be supplied: a description of the system's logical interconnect, the physical wafer description, and a map of the faulty cells and interconnect on the wafer. The logical interconnect description is currently "hardwired" as a combination of unidimensional/nearest-neighbor and broadcast signals. Pin identities are specified in an easily modified file and are input to the assigner and linkers. A description of the physical wafer is supplied through a wafer description language (see Appendix C). A data base that describes the wafer is constructed via this language. Several object types can be defined: a cell, channel, or array of links. An object type can be instantiated in the

data base by "putting" it on the wafer. After the data base is built, the linkers use data-base query operators to establish what interconnect resources are available. A map of bad resources is supplied externally through the testing process. Interconnect can be labeled "bad" by the use of data-base operators. The locations of functional cells are described in a separate file.

The assigner was adapted from previous work and works well for sufficiently high cell yields. Instead of simply mapping logical cells to good physical cells, it attempts to map logical columns of cells to columns of good physical cells. If there are any defective cells in the physical column, the assigner searches in a progressively widening spiral for the closest good cell. This spiral search exhibits an inherent bias toward a downward direction which tends to keep the mapping of logical columns to physical columns as close as possible. Figure 3(a) gives an indication of the spiral search procedure, and Fig. 3(b) is an example of its behavior for a particular defect pattern.

The two linkers which are included handle two different kinds of signals. The comb linker is used for broadcast signals. The term "comb" refers to the way the signal would be routed on the rectilinear interconnect, resembling the teeth of a comb. The comb linker would normally be operated first to reserve channel tracks for the broadcast signals. The 2-pin net linker, however, serves the daisy-chained signals running from cell to cell. This linker matches the connectivity requirement against a predetermined library of patterns to determine what the linking protocol should be. The pattern selected depends on the location of the cells relative to each other, and on the sides of the cells that the connection points of interest are located. For the integrator, the number of unique patterns possible is quite limited, which makes an exhaustive library search tractable.

The low-level table primitives that control the laser and table assemblies have been implemented. The VAX and the laser controller computer are connected by an RS-232 line such that command files generated by the linkers can be transferred from the VAX to the laser controller. Over this line the controller then sequences through the commands, invoking the low-level primitives that alter the wafer interconnect as it proceeds.

C. YIELD VS REDUNDANCY FOR ARBITRARY SEGMENTATION

It is intuitive that the availability of arbitrary bus segmentation offered by the laser link technology will reduce the amount of interconnect required without sacrificing system yield. In order to provide guidelines to designers, yield-vs-redundancy behavior for a given interconnect capacity must be quantified under the premise of this new technology.

In the previous semiannual,[4] a cell assignment procedure called BISKIP was described. The BISKIP assigner was designed for logical arrays of unidimensional next-neighbor connected, identical cells. In this strategy, surplus cells from adjacent columns can be assigned to another logical column. This scheme is parameterized as BISKIP(K, L), where K is the maximum vertical skip permitted and L is the horizontal. This procedure requires two L + 2 horizontal tracks and two K + 2 vertical tracks per signal. Simulations of wafers with a 50-percent cell yield were performed where the X and Y sizes of the wafers and the K and L parameters were varied. However, the BISKIP algorithm was predicated on the assumption of fixed segmentation and a certain amount of physical interconnect being available.

A technique similar to an unlimited BISKIP has been developed and can be implemented efficiently given an arbitrary bus segmentation capability with two horizontal and two vertical tracks per signal. The restriction of this method is that cells of two logical channels cannot overlap in a physical channel. Simulations of this arbitrary segmentation BISKIP were performed using an exhaustive assignment procedure. Assignments of a simulated wafer with 50-percent defective cells were permuted until a guaranteed linkable assignment was generated or all possible assignments were exhausted. Figure 4 shows a plot of the relative system yields of the arbitrary segmentation BISKIP, BISKIP(1,2), and the theoretically maximum system yield (unconstrained assignment). Also shown is an "unlimited skip" assigner which can operate unconstrained within a given column, but cannot make use of resources outside of that column. As can be seen, the arbitrary segmentation BISKIP operates close to the theoretical maximum system yield vs redundancy curve. Therefore, this assigner has been adopted for use in an arbitrary segmentation environment.

The preliminary phase of the arbitrary segmentation BISKIP assignment procedure amounts to the spiral-search approach of the previous section. If cell yields are sufficiently high, then a more "intelligent" assignment procedure than the simple spiral search may not be warranted. If cell yields are lower, it will be necessary to invoke the full arbitrary segmentation BISKIP assigner. Experience with actual processed integrator wafers will be required before a final decision can be reached.

D. DESIGN RULE CHECKING AND CALMA/CIF INTERFACE

A sophisticated design-rule checking system is a necessity in today's world of IC design to ensure integrity of layout. Other methods such as visual inspection of masks and checkplots, and iterating through fabrication are both inadequate and undesirable.

The Mask Design Rule Checking (MDRC) system,[5] which was originally developed under a companion Air Force program, has several features which make it an extremely powerful tool. The MDRC system is efficient, has an extremely low false-alarm rate, is technology independent, and has a high-level user interface.

The MDRC system comprises two modules: a Mask Processing Machine (MPM), and a macro translator. The MPM executes low-level mask instructions (software emulation) which are categorized as follows: preprocessing, spacing, logical, topological, and input/output. The macro translator provides a high-level user interface to facilitate the writing of programs for the MPM. This is a two-pass translator which performs a macro expansion and a syntax check. The user may draw from a predefined library or define his own macros.

Work is nearly complete on transferring the MDRC system to the VAX machine. It was originally coded in PL/I on an Amdahl computer. Since there is no PL/I compiler available on VAX/UNIX, the system is being recoded in the C language. This recoding process is nearing completion, and currently some 23 MPM primitives have been implemented on VAX/UNIX. This comprises enough of the system so that we have been able to perform operations such as intersection, complement, and overlap check on actual mask data. Preliminary indications are that run times will be satisfactory.

In addition, MDRC (VAX) has primitives to read and write CIF. When fully operational, MDRC (VAX) will provide a very flexible rules-checking capability. When coupled with a data standard such as CIF and a processing standard such as MOSIS, MDRC could provide a powerful network service. Remote users could send CIF descriptions of chips over the DARPA network, and CIF descriptions of errors would be returned.

The Calma/CIF interface is now operational. This interface takes the form of Mann-CIF and CIF-Mann translation routines on the VAX. Mann-CIF provides the designer with the capability of taking cells designed on the Calma and transferring them to the VAX to form a CIF library. CIF-Mann enables the designer to perform graphic editing on designs originating on the VAX, or use some other Calma program such as pattern generator fracture. It is expected that this interface will be very useful in the future.

E. YIELD MODELING AND INTERCONNECT THEORY

An effort has been initiated to develop a theory of device cost based on active area, redundancy, yield, interconnect area, and algorithm execution time. Previous work has dealt only with algorithm execution time vs active area. Recently, the effect of interconnect area has been acknowledged, but little systematic work has been done. The application of redundancy and yield to the other measures of cost is relatively new.

1. Interconnect

The concept of an Area Utilization Ratio (AUR) has been developed which is defined as:

$$AUR = \frac{A_i}{A_c}$$

where

A_i = the total area of interconnect,

A_c = the total area of computation elements, and

$A = A_i + A_c$.

If $\lim_{A\to\infty} AUR = \infty$ for an interconnect scheme, then it becomes less-and-less desirable as technology makes it possible to pack more-and-more circuits onto a wafer. The interconnect area is consuming a greater-and-greater percentage of the system area, and at some point nearly all the increase in area is due to the interconnect. If $\lim_{A\to\infty} AUR = k$ (a constant) for an interconnect scheme, then it becomes more-and-more desirable as technology improves. An AUR that approaches infinity is termed "non-viable"; one that approaches a constant is considered "viable." It has been shown that several well-known interconnect strategies are non-viable including shuffle-exchange and butterfly switch networks.

To develop the theory, it is assumed that computation elements and interconnect are planar rather than three-dimensional, which is reasonable for the present state of VLSI technology. It is possible that it will soon be feasible to fabricate several levels of metal. Work is also being done toward three-dimensional interconnections of computational elements. The results described here for the planar case will still be valid for three dimensions of interconnect because the area in the computation plane needed for interconnect to descend from the n^{th} level of metal is proportional to n. This implies that the planar results need only be modified by a proportionality factor.

Some otherwise attractive system interconnect strategies fail the AUR test. Chief among these are the FFT (butterfly) network and the shuffle-exchange graph. As an example, AUR of the standard FFT network (as shown in Fig. 5) will be calculated. Assume that N is a power of 2. Then:

$$A_c = k_c N(1 + \log N) \quad .$$

The horizontal contribution to the interconnect area can be seen to be:

$$A_{ix} = k_{ix} \sum_{j=1}^{\log N} 2^j$$

$$= k_{ix}(2^{\log N+1} - 2)$$

$$= k_{ix}(2N - 2) \quad .$$

The vertical contribution to interconnect must be proportional to N. Therefore:

$$A_i = k_i(2N^2 - 2N)$$

and

$$\underset{A\to\infty}{AUR_{FFT}} = \frac{A_i}{A_c}$$

$$= \lim_{N\to\infty} \frac{k_i 2N}{k_c \log N}$$

$$= \infty \quad .$$

2. Yield Modeling

Two basic decisions face the designer of RVLSI systems. The appropriate size for the cells into which the system is partitioned must be chosen, and the appropriate number of spare cells must be determined. We term the first design parameter the "partitioning factor" which is actually an alternate formulation of the parameter, namely the number of cells required for the system to function rather than the cell size. The second parameter is called the "redundancy factor." These parameters should be chosen to minimize cost, where cost is defined as (system area)/(system yield). This has an interpretation as the average amount of silicon which must be processed to produce one system.

In the simplest mode where there is no redundancy, the system area is A and the system yield is $e^{\delta A}$, where δ is the average defect density. Next, the effect of redundancy on yield was added to the standard defect model. In this model, the system is partitioned into n identical cells. A redundant system is then constructed using rn cells, where $p > 1$ is the redundancy factor. Interconnect is assumed to be free, perfect, and sufficient. With this in mind, we see that system area is

$$Y(p,n,rn) = \sum_{i=n}^{rn} \binom{rn}{i} p^i (1-p)^{rn-i}$$

where p = cell yield = $e^{-\delta(A/n)}$.

Cost curves developed using this model demonstrate that there is an optimum redundancy factor which minimizes system cost (Fig. 6). Greater redundancy increases the area more than the yield and is detrimental to the cost. Surprisingly, the optimum redundancy factor is quite low, usually lower than 30 percent. However, because interconnect was ignored, an optimal value for the partitioning factor cannot be obtained from this model.

A more complicated model attempts to quantify the area required by interconnect. It does so by recursively partitioning the system into finer-and-finer components. At each level in the hierarchy, the pin count is calculated using an extension of Rent's rule.[6] Redundancy is then added at some level. The channel area needed to connect the components is then estimated. Adding this channel area into the system area gives a more accurate model of area which is dependent on the partitioning factor.

There are two reasons for this greater accuracy. First, the pin count of the level where redundancy is added is dependent on the partitioning factor, and thus affects the channel area needed to accommodate defect avoidance. Second, two different track widths are used to calculate channel width – one in levels below the redundant level, and a larger one for the redundant level and above. This larger track width reflects the actual size of the restructuring links, and the smaller width reflects the minimum metal line pitch. The details of this model are still in development; however, preliminary analytical results indicate that a minimum cost as well as optimum redundancy and partitioning factors can be found by this approach.

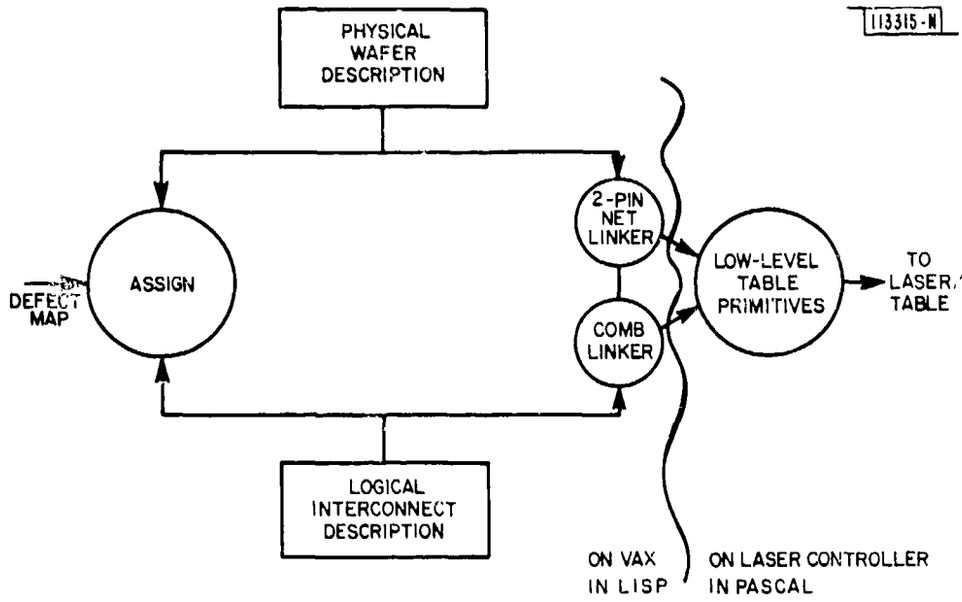
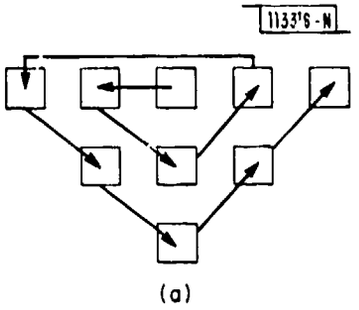


Fig. 2. Production-oriented assignment and linking.



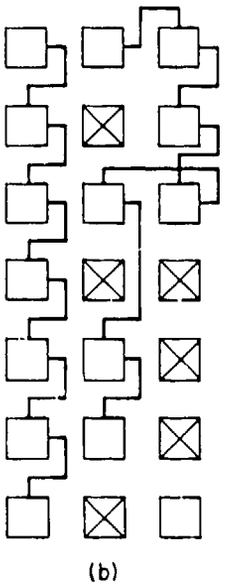


Fig. 3. Spiral search procedure: (a) search algorithm; (b) example for specific defect pattern.

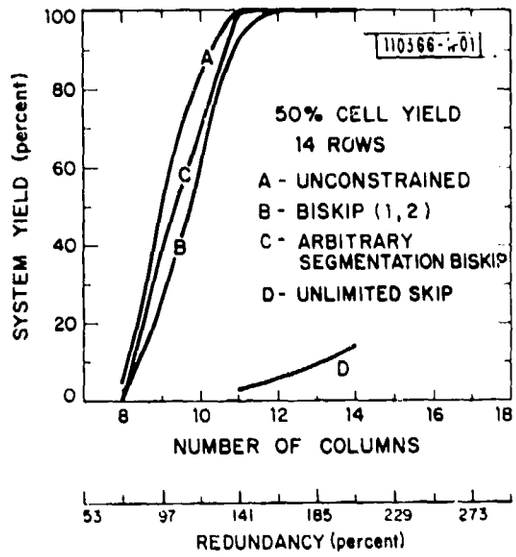


Fig. 4. Yield vs redundancy as function of assignment strategy.

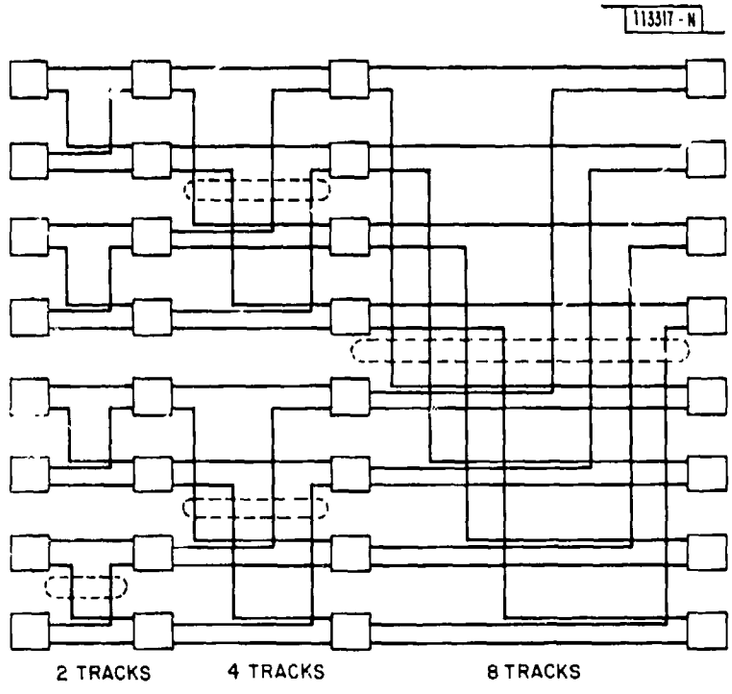


Fig. 5. FFT network.

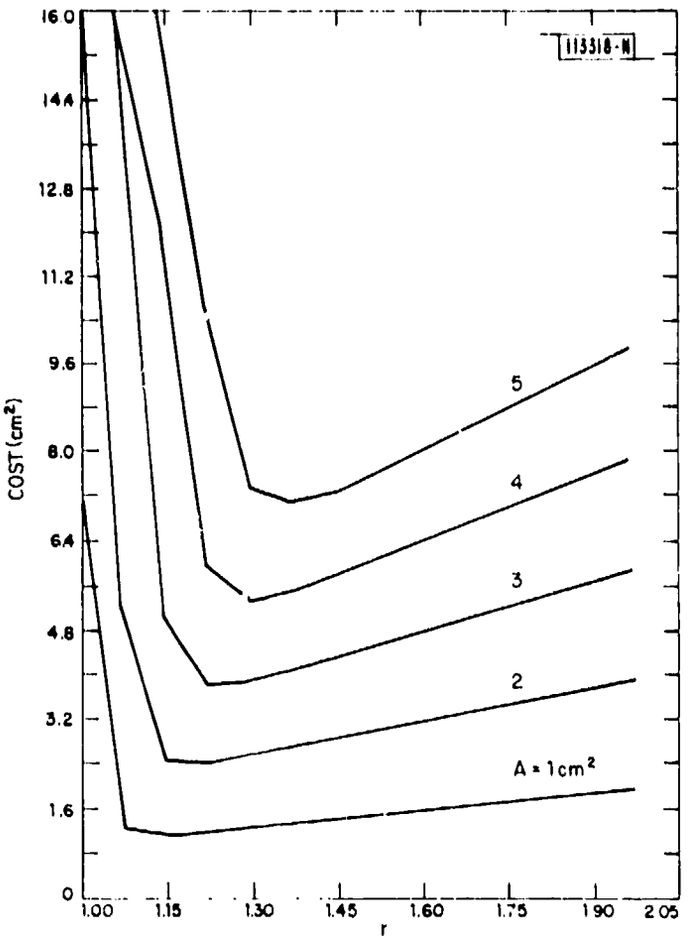


Fig. 6. Cost vs redundancy for n = 50 and $\delta = 2/cm^2$.

IV. TESTING AND APPLICATIONS

A. PHASE 0 INTEGRATOR

The Lincoln Bulk CMOS Process has been debugged and the Phase 0 Integrator wafers have been fabricated. Each wafer is a 7 × 7 array of 4-bit counters with an interconnect matrix of lines fully populated with laser links. The wafer-scale lines formed by putting images in the DSW photolithography process are of good quality. The cell yield is high enough to connect at least a 4 × 4 array and therefore test out assignment and linking algorithms and the physical process of cutting lines and making connections. A package sufficient for the current fabrication and testing sequences was designed, and several have been built. It is a ceramic plate about 2 in. square with a metal film pattern for fanning out from the chip to leads. Be-Cu lead frames are soldered to the four edges of the ceramic plate. Phase 0 wafers have been cut to a 1-in. square, fastened to the package, and wire bonded. The package is mounted on the laser table with connections to testing equipment which, at present, is standalone and not under computer control. The laser table is controlled by a minicomputer which has a data link to the VAX system. Laser linking experiments will be performed with these wafers starting in October 1981.

The functional tester has been modified to test the 4-bit counter cell. The unit was originally built for ECL circuits, thereby requiring an alteration in voltage levels. Counters on a preliminary CMOS test chip were successfully exercised and tested at a 1-MHz clock rate.

B. PHASE 1 INTEGRATOR

The Phase 1 Integrator combines a custom-designed 4 × 10 integrator cell with a full-wafer interconnect grid and laser-zappable interconnect pads. The cell also contains four bits of the high-speed input data shifter and output enable selector, ten internal data output tri-state buses, and ten tri-state drivers for transmitting data from the cell onto the interconnect grid. The transistor sizes have been calculated based on simulation studies, and the

cell layout has been completed and submitted for processing. Some effort has been directed toward the physical layout of the wafer and interconnect grid design, but this is as yet incomplete. A modification to the Mann 4800 Direct-Step-On-Wafer (DSW) machine has been specified which should greatly reduce the number of reticles required. First silicon for the basic cell is expected by the end of the first quarter in FY 82.

C. MOSIS TESTING

Process evaluation testing was performed on one MOSIS run during the last quarter, M18N. A different version of the test program was written for the new MOSIS test strip.

V. FACILITY STATUS

The SI disks have been installed on the VAX and are operational. The Berkeley UNIX (4.1 BSD) software has been installed and user files moved onto the new system. Some reorganization of the file system is under way to more effectively utilize the disk resources. The Berkeley CAESAR graphical layout system has been brought up on the VAX using a borrowed AED 512 terminal to verify compatibility with the local facility. A permanent unit is on order. All necessary hardware to implement a VAX ARPANET connection is in-house. Some minor problems with the port expander remain to be resolved. It will then be necessary to obtain the necessary drivers for the 4.1 BSD UNIX kernel as well as sundry utilities. The best means for acquiring this needed software are currently under investigation.

REFERENCES

1. J. Raffel et al., "Laser Programmed Vias for Restructurable VLSI," Proc. IEEE Intl. Electron Devices Mtg., Washington, D.C., 8-10 December 1980, pp. 132-135, DTIC AD-A102648.

2. A. Weinberger, "Large Scale Integration of MOS Complex Logic: A Layout Method," IEEE J. Solid-State Circuits SC-2, 182-189 (1967).

3. W. Plummer, "Chip Maker's Guide," Lincoln Laboratory internal memorandum (12 December 1980).

4. Semiannual Technical Summary, Restructurable VLSI Program, Lincoln Laboratory, M.I.T. (31 March 1981), DTIC AD-A108276.

5. A. J. Giovinazzo, "A Mask Design Rule Checking System," Proc. IEEE Intl. Conf. on Circuits and Computers ICCC '80, Port Chester, New York, 1-3 October 1980, pp. 932-936.

6. B. S. Landman and R. L. Russo, "On a Pin Versus Block Relationship For Partitions of Logic Graphs," IEEE Trans. Computers C-20, 1469-1479 (1971).

APPENDIX A
MACPITTS BNF

```
<program>          -> (program <program-name>
                       [<eval> | <def> | <always> | <process>]*)
<eval>             -> (eval [compile | simulate | both] LISP-FORM)
<def>              -> (def <macro-name> <type> LISP-FORM)
                   |  (def <register-name> register)
                   |  (def <port-name> port [input | output | i/o])
                   |  (def <signal-name> signal [input | output | i/o])
                   |  (def <constant-name> constant [CONSTANT | "CHARACTER"])
                   |  (def CONSTANT word-length)
                   |  (def <environ-name> environment LISP-FORM)
                   |  (def <organelle-name> organelle <gen-form> <sim-form>)
                   |  (def <function-name> function <#operands>
                        <organelle name> <control lines>)
                   |  (def <test-name> test <#operands>
                        <organelle-name> <control-lines> <test-lines>)
<type>             -> [process | form | formula | condition] [macro | list]
<gen-form>         -> LISP-FORM
<sim-form>         -> LISP-FORM
<#operands>        -> CONSTANT
<control-lines>    -> ([CONSTANT]*)
<test-lines>       -> ([CONSTANT]*)
<always>           -> (always [<form>]*)
<process>          -> (process <process name> <stack depth>
                        [<label> | <form>]*)
                   |  <macro>
<stack-depth>      -> CONSTANT
<label>            -> ATOM
<form>             -> (signal <signal-name>)
                   |  (go <label>)
                   |  (call <label>)
                   |  (return)
                   |  (cond [<guard>]*)
                   |  (setq <register-name> <formula>)
                   |  (setq <port-name> <formula>)
                   |  <macro>
<formula>          -> CONSTANT
                   |  "CHARACTER"
                   |  <constant-name>
                   |  <register-name>
                   |  <port-name>
                   |  (setq <register-name> <formula>)
                   |  (setq <port-name> <formula>)
                   |  (<function-name> [<formula>]*)
                   |  <macro>
<guard>            -> (<condition> [<form>]*)
<condition>        -> t
                   |  ()
                   |  <signal-name>
                   |  (and [<condition>]*)
                   |  (or [<condition>]*)
                   |  (not <condition>)
                   |  (bit <formula> CONSTANT)
                   |  (<test-name> [<formula>]*)
                   |  <macro>
<?-name>           -> ATOM
```

APPENDIX B

VAX/LASER CONTROLLER INTERFACE FORMAT

The proposed VAX - apple interface is as follows: All commands will be ASCII encoded. A command will begin with "Z", "T" , "P" , "X", "Y", "I", "J" or "G" and will terminate with a semi-colon ";".

The "Z" (zap) command has no operands.
The "T" (type) command causes the display of the remainder of the command. This is different from the "P" command, in that the apple will not pause.
The "P" (pause) causes the display of the remainder of the command. In addition the apple will pause in the execution of commands from the command file, and accept commands from the apple operator.
The "X" command causes a X direction delta move. The operand is the number of tables steps to move in the X direction.
The "Y" command causes a Y direction delta move.
The "I" command causes a Y direction "jog". The operand gives the number of jogs to be perfomed.
The "J" command causes a X direction "jog".
The "G" (goto) command causes the table to move to the location specified in the two operands. The operands are the absolute x-y coordinates in units of table-steps.

Any character (such as <lf> or <cr>) appearing between the semicolon and the next command header is ignored.

The BNF is as follows:

```
<command>      := {<zcom>|<tcom>|<pcom>|<xcom>|
                  <ycom>|<icom>|<jcom>|<gcom>};
<zcom>    := Z
<tcom>    := T<sp><string>
<pcom>    := P<sp><string>
<xcom>    := X<sp><number>
<ycom>    := Y<sp><number>
<icom>    := I<sp><number>
<jcom>    := J<sp><number>
<gcom>    := G<sp><number>,<number>
<sp>      := space
<string>  := any ASCII except ";"
<number>  := (<sp>*)(+|-)[<digit>*]
<digit>   := {0|1|2|3|4|5|6|7|8|9}
```

This format is an adaptation of the article "In-house Standards Fill Gaps in Instrument-computer Interface" Electronics 3/24/81.

APPENDIX C

BNF FOR WAFER DESCRIPTION LANGUAGE

```
auxilliary data base
     <link-descriptions>:= ([(<name> [<size> <use> <connect>]*)]*)
     <size>:= (size <xytu>)  ;size of link
     <xytu>:= (<ntu>)  ; x and y
     <ntu>:= number in xytable units
     <use>:= (use <n>)  ; number of times link can be programmed
     <connect>:= (connect <xytw>  ; coordinates to zap for connection

xyplane definitions
  links - definitions or link blocks.
     <links>:=([<name> (<size> <delta> <uses> <origin>)]*)
     <size>:= (size <xyn>)  ; number of links in linkblock
     <xyn>:= (<n>)          ; in x and y direction
     <delta>:= (delta <xytu>)  ;pitch of links in linkblock
     <uses>:= (uses <link-description-name>)  ;from auxilliary data-base
     <origin>:= (origin <xyn>)
  note:  links can be set in the xyplane definition table by using:
     (linkblockdef '(name <name>) ' <size> '<delta> '<uses> '<origin>)
  this is more likely to be correct than the actual bnf
cells - definitions of the physical aspects of cells
     <cells>:= ([<name> (<pins>)]*)
     <pins>:= (pins (<pin-description>*))
     <pin-description>:= (<pin=name> <delta-origin> <oriented>)
     <pin-name>:= (name <name>)
     <delta-origin>:= (delta-origin <xytu>)  ; location from center of cell
     <oriented>:= (oriented [l r b t])  ; left, right, bottom, or top
--channels - definitions of the physical aspects of channels
     <--channels>:= ([(<name> (<origin> <delta> <tracks>))]*)
     <origin>:= (origin <n>)
     <delta>:= (delta <ntu>)  ;pitch of tracks in channel
     <tracks>:= (tracks [()]*)  ;there are as many replications of ()
                                as there are tracks
```

GLOSSARY

AUR	Area Utilization Ratio
BNF	Baekus-Naur Form
CIF	Caltech Intermediate Form
CMOS	Complementary Metal-Oxide Semiconductor
DSW	Direct-Step-On-Wafer
FFT	Fast Fourier Transform
LICL	Lincoln Integrated Circuit Language
MDRC	Mask Design Rule Checking System
MNOS	Metal-Nitride-Oxide Semiconductor
MPM	Mask Processing Machine
PAL	Placement, Assignment, and Linking
PLA	Programmable Logic Array
RVLSI	Restructurable Very Large Scale Integration
VLSI	Very Large Scale Integration

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-81-331	2. GOVT ACCESSION NO. AD-A112037	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Restructurable VLSI Program		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 April – 30 September 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Peter E. Blankenship		8. CONTRACT OR GRANT NUMBER*(s)* F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 3797 Program Element No.61101E Project No. 1D30
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 30 September 1981
		13. NUMBER OF PAGES 42
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

VLSI	customization	routing
Restructurable VLSI (RVLSI)	hardware description language	systolic array
programmable interconnect	placement	integrator
defect avoidance		

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes work on the Restructurable VLSI Research Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the semiannual period 1 April through 30 September 1981.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*